How To Use DataProvider in TestNG – Parameterization

Automation Testing has evolved as a fixture in the software development process. No one wants to dedicate hours of human effort and time when they can obtain accurate results with suitable test scripts, tools, and frameworks.

This is why it is essential to learn the different characteristics of automated testing. TestNG comes up with DataProvider to automate providing test-cases for implementation.

DataProvider aids with data-driven test cases that hold the same processes but can be run multiple times with distinct data sets. It also helps in equipping complicated parameters to the test methods.

What is a Data-Driven Framework?

A data-driven framework caches the test data in a table or spreadsheet format. It permits automation engineers to use a single test script for accomplishing all test data present in the table. In the data-driven framework, input values are read from data files and held into a test script variable. Data-Driven testing allows the creation of both positive and negative test cases into a single test.

In this test automation network, input data can be stored in a single or multiple data source like XML, CSV, XLS, and databases.

TestNG Parameterization

Parameterization is an implementation strategy that runs a test case automatically, multiple times, with various input values. The test design comes with reading data from a file or database instead of the hard-coded values.

In TestNG, there are two methods to accomplish parameterization :

- With the aid of Parameters annotation and TestNG XML file @Parameters({“name,” “search key”})
- With the aid of DataProvider annotation @DataProvider(name= “searchProvider”)

TDataProvider in TestNG

First, to provide the test data, declare a method that returns the data set in the form of a two-dimensional object array Object.

The first array defines a data set, whereas the second array includes the parameter values. The DataProvider method can be in the identical test class or superclasses. It is also feasible to equip a DataProvider in another type, but the method search key is fixed.

After adding this method, interpret it using @DataProvider to let TestNG know it is a DataProvider method. Additionally, feed a name using the name attribute of the DataProvider annotation. If one hasn't provided the title, the name of the process will be used for reference by default.

Parameterization using DataProvider

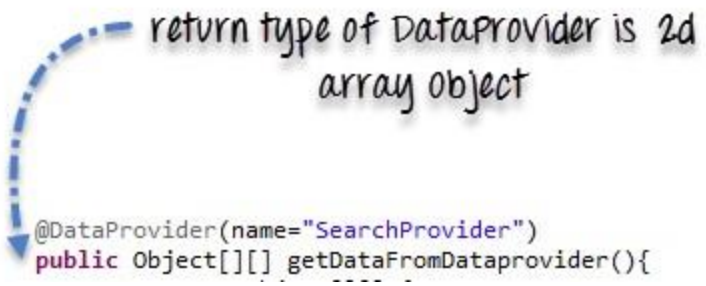
Documenting thousands of web forms using the testing framework is tiresome. To lessen this process, one needs a distinct process for conducting large datasets in a single implementation flow. This data-driven concept is accomplished through @DataProvider annotation in the TestNG framework.

It has only a single attribute, 'name.' If one does not define the name attribute, the DataProvider's name will be identical to the

corresponding process name. This is how DataProvider facilitates the task of testing multiple sets of data.

Let us take an example to explain how it works.

In this example, the tester desires to automate entering username and password and log into twitter.com. Therefore, this test case should run two times with distinct sets of data (the data provided in the 2D array).



Let's execute the same and see how it functions.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.Chrome.ChromeDriver;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
public class DataProviderTest {
// this will take data from dataprovider which we created
@Test(dataProvider="testdata")
public void TestChrome(String uname,String password){
System.setProperty("webdriver.chrome.driver", "Path of the driver");
WebDriver driver = new ChromeDriver();
// Maximize browser
driver.manage().window().maximize();
// Load application
driver.get("https://twitter.com/login");
// clear email field
driver.findElement(By.name("session[username_or_email]")).clear()
;
```

```
// Enter username
driver.findElement(By.name("session[username_or_email]")).sendK
eys(uname);
// Clear password field
driver.findElement(By.name("session[password]")).clear();
// Enter password
driver.findElement(By.name("session[password]")).sendKeys(passwo
rd);
}
@DataProvider(name="testdata")
public Object[][] TestDataFeed(){

// Create object array with two rows and two columns- the first
parameter is row and second is //column
Object [][] twitter data=new Object[2][2];

// Enter data to row 0 column 0
twitterdata[0][0]="username1@gmail.com";
// Enter data to row 0 column 1
twitter data[0][1]="1password";
// Enter data to row 1 column 0
twitterdata[1][0]="username2@gmail.com";
// Enter data to row 1 column 0
twitter data[1][1]="Password2";
// return arrayobject to test script
return twitter data;
}
}
```

The driver will launch Chrome on executing this program, navigate to the Twitter home page, and enter the defined username and password combinations.

DataProvider can also enter data combinations across multiple websites simultaneously.

Conclusion

Running Selenium tests employing DataProvider and TestNG is ideal for speeding up test cycles, demonstrating more comprehensive automated testing of websites, and making outstanding user experiences with minimal time, effort, and resources. Therefore, it should feature in testing pipelines, making testers' lives infinitely more accessible. But, of course, these tests are always best to run on real browsers and devices.